**Partea 1 (4p --- cica, nu stiu daca este asa cu adevarat)**

**1. (cu cache, teorie si probleme)**

* **Explicati cum sunt afectate cele 3 tipuri de miss (obligatoriu, de conflict si de capacitate) daca adaugam capabilitate de prefetching unui cache. (08.09.2014) (03.02.2014)**

--- **Miss obligatoriu scade, pentru ca sunt aduse dinainte datele.**

--- **Miss de capacitate scade**, **(in special daca face prefetch la ce trebuie si cand trebuie):** ideea e ca eu ma gandesc ca miss-ul de capacitate are loc atunci cand are loc miss indiferent de politica de inlocuire - daca eu am o politica de inlocuire perfecta si trebuie sa aduc 100MB intr-un cache de 10MB, la un moment dat tot o sa am miss, dar daca sa zicem ca nu folosesc toti 100MB unii dupa altii, e posibil sa am timp sa fac prefetch la ceva ce urmeaza si sa scap de miss-ul de capacitate.

**-- Miss de capacitate -> scade. Idem Tibi. Daca avem nevoie de mai multa informatie decat poate cache-ul sa tina, tehnica de prefetch poate sa aduca informatia pe care nu am reusit sa o incarcam in cache atunci cand este nevoie.**

--- **Miss de capacitate:** eu aici cred ca nu se modifica deoarece cache-ul are aceeasi capacitate si daca eu aduc niste date de care am nevoie mai devreme, inlocuind in acelasi timp alte date utile, asta nu inseamna ca per total missurile de capacitate vor scadea, ci se vor petrece mai devreme decat s-ar fi petrecut in mod obisnuit. (mie mi se pare normal sa ramana la fel, desi inteleg ce zici, Tibi, dar capacitatea cache-ului ramane aceeasi si eu cand fac prefetch doar inlocuiesc datele din cache cu alte date, cum a zis Miruna, deci mi se pare ca folosesc cache-ul cum se foloseste in mod normal si ca nu se aduce nicio imbunatatire)

--- **Miss de conflict ar putea scadea in cazul in care valorile care sunt mapate in aceeasi locatie din cache nu sunt folosite atat de rapid una dupa alta, incat sa nu aiba timp prefetch-ul sa le inlocuiasca atunci cand e nevoie de ele. In schimb, daca ele sunt folosite una dupa alta, prefetch-ul nu imbunatateste cu nimic viteza, deoarece nu se poate face prefecth intr-o locatie care e folosita in mod curent.**

* **Daca ai linii de 2 ori mai lungi in cache spune ce se intampla cu miss obligatoriu, miss de conflict si miss de capacitate (28.01.2014)**

Ok, deci aici dimensiunea ramane constanta iar ele sunt de doua ori mai lungi.

---Deci in cazul asta scad miss urile obligatorii pentru ca aducem mai multe date,

---scad miss urile de capacitate (creste dimensiune cache). ??? nu zisesem ca dimensiunea cache ramane aceeasi? (acum am vazut ca in curs spune ca scad la reload (ce o mai fi insemnand si aia), si ca a ramas in dubiu si cu Dan Tudose de ce ele scad) aici ma gandesc ca e posibil sa fie acelasi motiv pentru care scad miss-urile de obligatorii si anume ca se aduc mai multe date => scade nevoia de a tot inlocui date in cache cu unele din RAM => mai putine sanse de a da miss.

---Aici in curs scrie ca CRESC miss urile de conflict ( scuze :”>).

Eu zic ca miss-ul de capacitate nu scade.(daca nu se umbla la dimensiunea chache-ului, nu cred ca se afecteaza miss-urile de capacitate) Ar putea sa creasca in cazul in care trebuie accesate adresele de forma 10k unde liniile au dimensiune 9. ca la un moment dat se umple cache-ul de linii si trebuie invalidate, dar daca liniile erau doar de un octet se umplea doar cu cate un octet, nu cu cate 9 si aveau loc toate. (totusi, cele de capacitate parca erau influentate strict de dimensiunea cache-ului. Daca eu cer adrese care imi dezorganizeaza cache-ul, nu cred ca e de capacitate)

* **Cum sunt afectati parametrii de hit time, miss rate, miss penalty daca adaugam capabilitate de prefetching unui cache? (10.02.2015)**

--- hit time ramane la fel

--- scad miss rate (1 - hit rate)

--- miss penalty ramane constant

* **Cum sunt afectati parametrii de hit time, miss rate, miss penalty daca dublam asociativitatea unui cash set-asociativ (dar pastram capacitatea si dimensiunea liniilor constante)? (08.09.2014) (03.02.2014)**

Tinda sa reduca miss uri conflictuale deci scad miss rate, creste hit time , creste miss penalty ( explicatie Tibi)

ideea e ca un set are acum mai multe linii daca creste asociativitatea

full associative inseamna un set care cuprinde toate liniile, adica toate pot fi mapate oriunde

asta daca folosim "set" asa cum e el denumit de fapt, nu cum a zis Dan Tudose la curs, cand se referea la set ca locul in cadrul unui index

--- creste hit time pentru ca ii ia mai mult sa caute

--- scade miss rate pentru ca scad miss-urile de conflict (bine-nteles trebuie o politica buna)

---- miss penalty creste pt ca adaugi cautarea + inlocuirea (mai faci si politica sa vezi pe care o inlocuiesti, sunt mai multe locuri in care poti inlocui) dar politica de inlocuire exista de la inceput si daca este una perfecta, atunci nu ar trebui sa dureze mai mult pana inlocuieste datele, nu?

* **Daca ai linii de 2 ori mai scurte (cache e la jumate din capacitate pt ca se pastreaza asociativitatea si inca ceva, probabil dimensiunea liniilor) ce se intampla cu hit time, miss rate si miss penalty (28.01.2014)**
* hit time: scade --------> pentru ca dureaza mai putin sa aduci datele din cache (eu aici as fi zis ca ramane la fel, ca pana la urma din cache cred ca se ia doar ce date se dorec din linia aia) si eu zic tot ca ramane la fel
* miss rate: creste ----------> mai putine date in cache
* miss penalty: scade ---------> (cred ca aici scade miss penalty deoarece se aduc din RAM calupuri de date mai mici, deci se ocupa mai putina latime de banda pe magistrala, timp de transfer mai scurt, probabil) (si eu cred ca scade, ia mai putin timp sa aduci datele din RAM) si eu zic tot ca scade
* miss de capacitate: creste
* miss obligatoriu: la fel creste ca sunt aduse mai putine instructiuni si deci trebuie facute mai multe accese la memorie obligatorii
* miss de conflict: scade? creste ca sunt mai multe linii in RAM de mapat, deci concureaza mai multe pe acelasi loc in cache

------------------------------------------------------------------------------------------------------------------------

Inainte de probleme, poate ar fi bine sa scriem niste formule de referinta (daca sunt gresite, yell and modify :) )

**Structura : tag index offset**

* **dimensiune linie cache = 2 ^ offset**
* **dimensiune cache = 2 ^ index linii**
* **dimensiune cache = 2 ^ offset \* 2 ^ index \* nr\_seturi**

**= 2 ^ (offset + index) \* nr\_seturi**

* **Avem o memorie cache cu cuvinte de un octet, linii de 8 octeti lungime organizata in 32 de seturi de cate 4 linii. Care este dimensiunea utila a memoriei cache, in octeti? Daca adresarea se face pe 32 de biti, care este dimensiunea tag-ului pt. fiecare linie de cache? (10.02.2015)**

(

fiecare linie are 8 octeti | fiecare set are 4 linii => 4\*8 octeti = 32 octeti | sunt 32 de set-uri => 32\*32 = 1024 octeti dimensiune totala cache

linii de 8 octeti => adresare pe 3 biti. (offset)

32 de seturi => adresare pe 5 biti (index)

=> tag-ul e de 24 de biti

)

* **Care sunt componentele unei adrese pe 16 biti a unui cache mapat direct, cu datele accesibile la nivel de octet. Linia de cache are o latime de W = 4 octeti. Dimensiunea cache L = 1024 linii (4kB)? (08.09.2014)** (28.01.2014)

Structura : tag index offset

W = 4 octeti = 4 \* 8 biti = 32 biti

latime\_linie = 2 ^ offset = 32 => offset = log2(32) = 5 biti pentru offset (aici datele sunt accesibile al nivel de octet => calcul cu cei 4 octeti, nu mai transformi in biti => 2 biti pt offset) da, 2 biti aici pt offset

L = 1024 linii = 2 ^ index => index = log2(1024) = 10 biti index

16 - 10 - 5 = 1 bit pentru tag (si aici cred ca ar veni: 4 biti pt tag ramasi) si eu cred ca rezultatul e 4b pt tag (model: slide 30, curs 2)

**Raspuns: (2 biti offset, 10 biti index, 4 biti tag)**

* **Avem un sistem de calcul cu un singur nivel de cache (L1). Daca adaugam un al doilea nivel de cache (L2), cum se vor modifica urmatorii parametri de functionare: L1 hit time, L1 miss rate, L1 miss penalty? (03.02.2014)**

Scade miss penalty ( mai mic pentru a aduce date din L2 ),

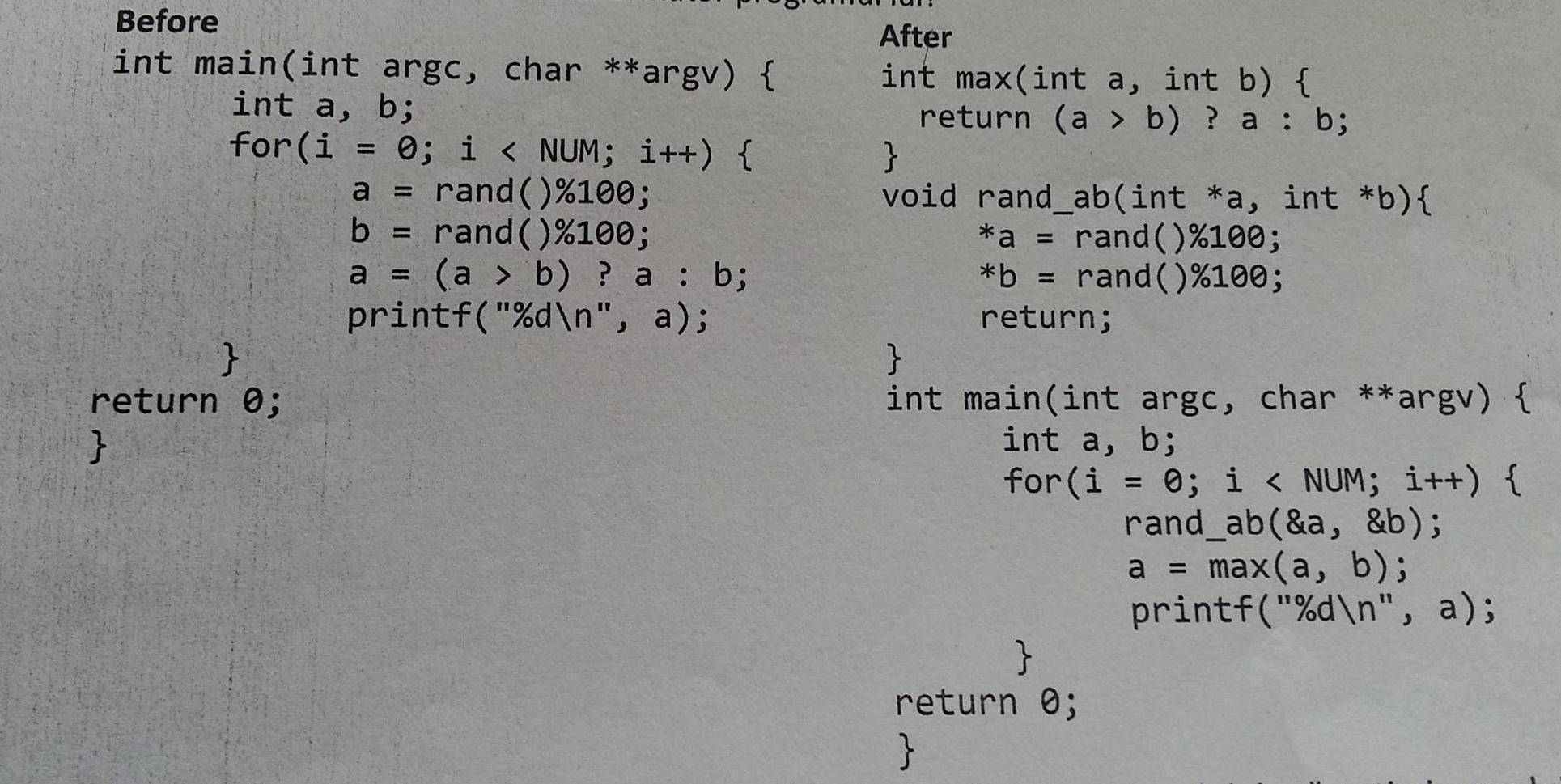
miss si hit raman la fel ( nu pot aduce mai multe date sau modifica dimensiune deci…)

**In curs scrie asa: (curs 3, slide 33):**

* **hit time mult mai mic (pentru ca prezenta lui L2 duce la un L1 mai mic) buna observatie, nu m-as fi gandit - dar trebuie specificat ca schimbi L1 daca zici asta**
* **miss rate mai mare (pentru ca L1 e mai mic)**
* **miss penalty mai mic pt L1 (nu scrie de ce, dar probabil pentru ca suporta L2 miss penalty-ul)**
* **Care din urmatoarele sunt adevarate? (10.02.2015)**
  + **un cache set-asociativ cu o linie pe set este un cache complet asociativ**
  + **un cache set-asociativ cu un singur set este un cache complet asociativ da da da oui** да
  + **un cache set-asociativ cu o linie pe set este un cache mapat direct da da da oui** да
  + **un cache set-asociativ cu un singur set este un cache mapat direct**

**2. (cu cod imbunatatit sau nu dupa modificare)**

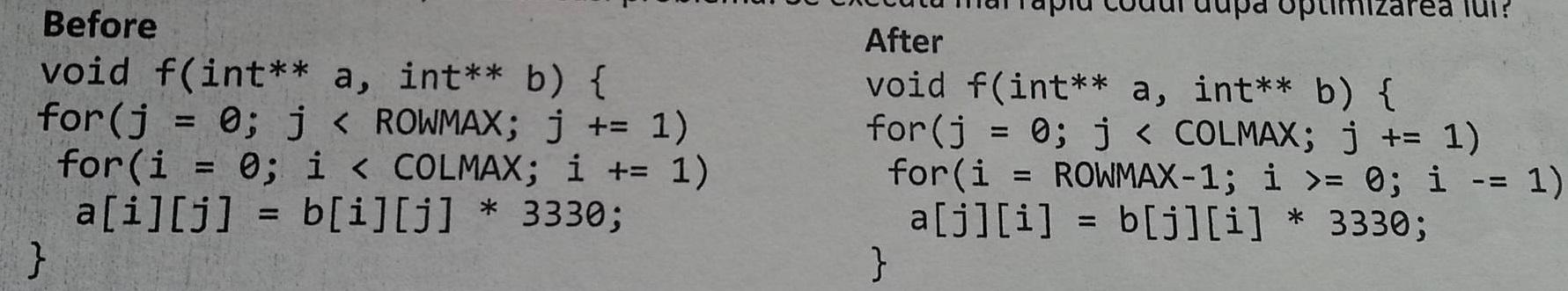
* **Se modifica cod pentru a functiona mai repede. Este ok? (10.02.2015)**



Nu (scade localitatea spatiala) (nu este vorba de localitatea temporala?: gen a si b din primul exemplu sunt tot folosite, pe cand in exemplul 2 se aloca alti pointeri de a si de b, cred), si pe de alta parte mai face si apeluri de stiva multe pt chemare functiilor alora => ineficient

(Cred ca accentul trebuie sa fie pus pe apelul de stiva, nu cred ca se refera neaparat la localizare temporala/spatiala) asa e, nu avem vectori, deci scade localitatea temporala

* **Se modifica cod pentru a functiona mai repede. Este ok? (10.02.2015)**

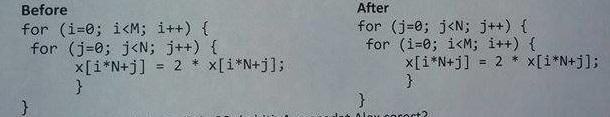


Nu(prima foloseste localitate spatiala). In memoria matricele sunt reprezinta ca o zona contigua de memorie. de fapt e cum zice Andrei mai jos, a doua are localitatea sptiala mai buna

(eu pe asta nu o inteleg … adica fooseste j drept contor de linii, dar in a il scrie asa: a[i][j] … adica drept index de coloane …. chestia asta da segfault daca i si j sunt diferite … gresesc?) Nu observasem asta, dar desi trece de bariera logica a notatiilor, sunt totusi niste variabile, deci poate sa le ia si sa le numeasca cum vrea el.

**Le numeste asa ca sa te induca in eroare. Pe primul parcurg matricea coloana cu coloana (ceea ce e naspa). In al doilea, parcurg rand cu rand, dar randurile le parcurg invers. Eu cred ca se imbunatateste localitatea spatiala, deci aduce o imbunatatire.**

* **Se modifica cod pentru a functiona mai repede. Este ok? (08.09.2014)**



Identic cu cel anterior? cred ca da

eu cred ca aici al doilea este mai bun la localitate temporala … adica in bucla din interior dubleaza de M ori aceeasi valoare din vectorul X, pe cand in primul face dublarea la intervale de N operatii;

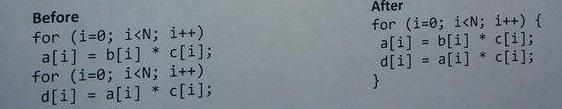
(eu cred ca nu se aduce o imbunatarie, la primul, parcurgerea se face pe randuri, la al doilea pe coloane => localizare spatiala mai proasta la al doilea => NU este ok)

(pai, Andrei, nu parcurgi o matrice, ci un vector …. adica nu ai coloane si linii; M-ul ala este acolo sa indice de cate ori inmultesti cu 2, nu?)

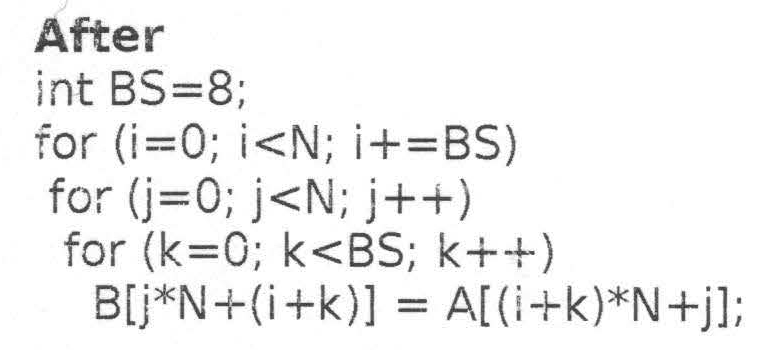
**Primul are localitate spatiala mai buna pentru ca in vector creste cu cate o unitate la fiecare iteratie, nu cu cate N**

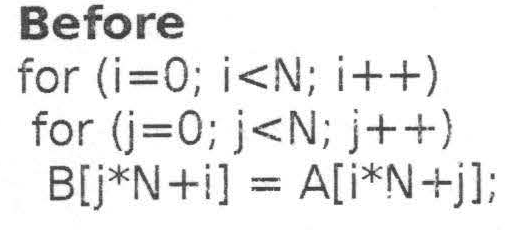
**Subscriu ^^^^**

* **Se modifica cod pentru a functiona mai repede. Este ok? (08.09.2014)**



A doua varianta mai buna. In primul caz va invalida “a” si eventual actualiza in RAM (rau). In al doilea caz il foloseste din cache. de acord se fol astfel localitatea temporala a[i] si c[i] folosite de mai multe ori да

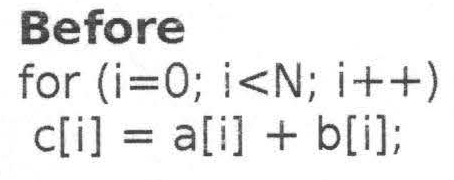
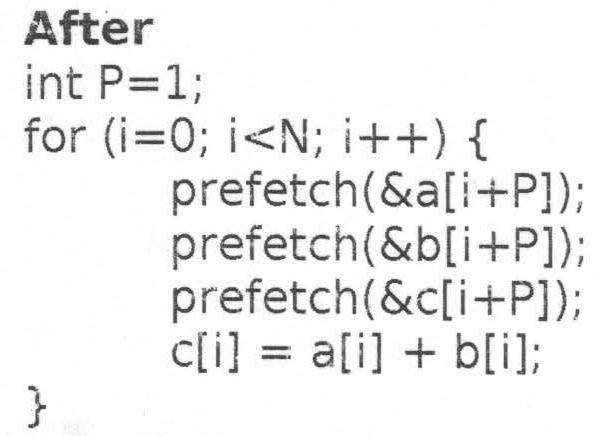
* **Se modifica cod pentru a functiona mai repede. Este ok? (28.01.2014, dar alt cod) (03.02.2014)**



Prima varianta foloseste localitate spatiala? Se pare ca, luand in considerare documentul pus de Tibi <http://csapp.cs.cmu.edu/2e/waside/waside-blocking.pdf>, a doua varianta prezinta localitate temporala, so functioneaza mai repede. Si in general, orice e organizat pe blocuri prezinta o mai buna localitate temporala.

(primul e clar ca nu merge bine pentru localitatea spatiala -> al doilea e pe blocuri -> al doilea e mai bun) algoritmul de aici nu e identic cu cel pe blocuri, dar e un pas spre ele deci are localitatea temporala mai buna

* **Se modifica cod pentru a functiona mai repede. Este ok? (28.01.2014, dar alt cod) (03.02.2014)**



**Mai buna prima varianta. Este prea tarziu pentru prefech si va polua cache ul. Spunea Dan ca era ok daca ar fi fost facut prefetch cu ~100 instructiuni inainte.** nu imi amintesc de asta cu 100 instructiuni, dar si eu mi-am notat ca nu e buna a doua varianta.

Am notat eu asta pe curs :? -> oricum mai buna prima da, clar

(prefetch-ul e costisitor, nu? Si daca eu fac prefetch la un element, e clar ca nu se va aduce doar acel element in cache, se va aduce intreaga linie, deci urmatoarele prefetch-uri sunt degeaba. Dar daca doar maresc P-ul, de ce ar fi mai bine?)

3. **(memorie virtuala, TLB: teorie sau probleme)**

* **Daca ai pagini de 2 ori mai mari, ce se intampla cu rata de miss si cu TLB reach (28.01.2014)**

Miss rate banuiesc ca scade, avand in vedere ca avem mai multa informatie la dispozitie => sanse mai mici sa dam miss

TLB reach este direct proportional cu dimensiunea paginii, deci si el creste:

TLB reach = numar\_intrari\_TLB \* dimensiune\_pagina \* numar\_pagini\_pe\_intrare? sincer nu stiu, in curs era doar prima parte

Am si eu o intreabare aici. TLB Reach spune ca este dimensiunea celui mai mare spatiu care poate fi mapat SIMULTAN. Deci daca am fi crescut doar numarul de intrari am fi fost siguri ca este contiguu si ar fi crescut TLB Reach?

* **Avem un sistem cu memorie virtuala si TLB. Daca dublam numarul de intrari in TLB, cum vor fi afectate rata de miss a TLB si capacitatea acestuia de adresare (TLB reach)? (08.09.2014) (03.02.2014)**

Rata de miss scade.

La fel ca mai sus (dimensiune pagina = numar de intrari)

Ah , este chiat aici intreabarea. Daca nu este contiguu nu cred ca putem spune sigur de TLB Reach. Gresesc? !!!! Buna observatie, Lavinia, nici eu nu stiu cum este… :( asa e, in curs scria ca pt memorie contigua, deci nu stiu ce se intampla daca nu e asa

* **Descrieti rolul pe care il are TLB intr-o arhitectura de calcul moderrna. (10.02.2015) (03.02.2014)**

TLB (Translation Lookaside Buffer) este un cache mic care contine cele mai recente translatii adresa virtuala - adresa fizica. Are rolul de a reduce numarul de accese la memorie pentru majoritatea operatiilor load/store (de la 2 cicli la 1).

* **Ati scris un program care are nevoie frecvent de cativa kB de memorie temporara. Ce este mai rapid: sa fol in mod repetat acelasi spatiu de memorie sau sa alocam un spatiu diferit de fiecare data? De ce? (10.02.2015)**

Daca datele de intrare nu difera de la o rulare la alta, atunci e mai rapid sa folosim acelasi spatiu de memorie, pentru ca datele sunt deja in cache si cum stim deja, e mai rapid sa accesam datele din cache decat din RAM sau HDD.

Eu as zice ca mereu e mai bine sa folosim in mod repetat acelasi spatiu pentru ca imbunatateste localitatea temporala.

* **Care dintre urmatoarele sunt motive valide pt. fol. adreselor virtuale? Explicati. (10.02.2015)**
  + **permite unui proces sa acceseze intregul spatiu de memorie** ---- Da, in sensul ca ii lasa impresia procesului ca are acces la toata memoria disponibila Nu, tocmai ca e limitat in spatiul pe care i-l pune la dispozitie sistemul de operare ??? (si eu as zice ca Alina … ca macar ii da impresia ca lucreaza cu toata memoria) e cum spui, Tibi, el e spatiu limitat, dar procesul crede ca lucreaza cu toata memoria
  + mareste rata de transfer a dispozitivelor I/O tind sa spun “nu”, de ce ar fi asa? (I/O-ul nu foloseste VM, nu?) Foloseste….in sensul in care umbla cu RAM-ul si HDD-ul prin DMA, dar cred ca adresele virtuale ii ingreuneaza treba … pt ca el in esenta cred ca lucreaza cu adrese fizice Nu mareste rata de transfer, poate chiar aduce un overhead prin operatii de translatare de adrese
  + **este mai dificil ca un program ce functioneaza prost sa afecteze alte programe concurente** -------- Da, pentru ca fiecare proces are propriul spatiu de memorie virtuala
  + **simplifica implementarea functiilor de memorie (malloc, new)** (unul ditre principalele scopuri) eu cred ca asta este “da”, pt ca scrie prin curs ceva de genul ca inainte de adrese virtuale, cei care scriau programe trebuiau sa fie atenti ei la alocari de memorie sa nu intre peste alte programe imi amintesc si eu ca scria asta da, simplifica pentru ca vede tot spatiul programului ca un spatiu contiguu de adrese si ii e mai usor sa il gestioneze decat daca ar avea la dispozitie toata memoria
  + **permite upgrade-ul gratuit de al Win7 la Win10 ------- clar da, nici nu exista dubiu, daca accentul e pus pe gratuit clar nu (daca nu era memorie virtuala, managementul memoriei era foarte dificil si dependend de arhitectura. Acuma, daca accentul e pus pe “gratuit” - e o prostie, asta tine de Windows, deci nu. Dar, daca accentul e pus pe “permite”, poate si memoria virtuala sa contribuie la chestia asta si as zice “da” :) )** si eu as zice tot ca da, desi nu prea are sens :D daca accentul pica pe gratuit, atunci sigur e corect, ca fiind adrese virtuale costa mai putin, adica e gratuit.

**4. (cu magistrale sau ce mai este pe la I/O)**

* **La care din urmatoarele magistrale este legat CPU-ul? (10.02.2015)**

1. **CPU bus; b) Memory bus; c) School bus d) I/O bus e) PCI bus**

Alina: a, b asta este dubioasa pt ca in curs scrie magistrala procesor-memorie … nu este scris niciodata separat … cred ca merita intrebata daca ne pica la examen exact, stiu, de-asta le-am ales pe amandoua :(

* **Dati 3 exemple la alegere de magistrale dintr-un sistem de calcul si descrieti pe scurt la ce sunt folosite. (08.09.2014) (28.01.2014)**

1. Magistrala Procesor-Memorie:
   1. conectata direct la CPU;
   2. conecteaza sistemul de memorie;
   3. maximizeaza bandwidth-ul Memorie - CPU
   4. optimizata pentru transferuri de blocuri cache
   5. scurta si de viteza mare (detaliu)
2. Magistrala I/O:
   1. trebuie sa acomodeze o gama larga de dispozitive I/O
   2. lunga si lenta (detaliu)
   3. se conecteaza la magistrala procesor-memorie sau la backplane (detaliu)
3. Backplane bus:
   1. permite procesoarelor, dispozitivelor de I/O si memoriei sa coexiste
   2. o structura de interconectare din interiorul sasiului (detaliu)

4. posibil sa mearga povestit si despre PCI

* **In sistemele de calcul moderne se folosesc din ce in ce mai mult magistrale de comunicatie seriala (SATA, USB, etc.). De ce sunt acestea preferate magistralelor paralele (dati cel putin 3 motive)? (03.02.2014)**
* functioneaza la viteze multi-gigabit
* low power => avem o singura legatura de date per periferic
* transferuri simultane multiple
* conectori si cabluri ieftine
* fiecare dispozitiv are legatura lui de date
* **Descrieti, pe scurt, modul de functionare al unui controller DMA (asta este de la I/O). (08.09.2014) (28.01.2014)**

DMA = direct memory access

Pasii de functionare sunt:

1. un dispozitiv cere transferul a S octeti de la disk controller; specifica adresa A si dimensiunea S controllerului DMA;

2. disk controller incepe transferul DMA a S octeti;

3. disk controller initiaza bus request pentru fiecare octet de pe disk;

4. controller DMA transfera cate un octet la adresa A, incrementeaza A si decrementeaza S;

5. cand S == 0, controller DMA intrerupe CPU pt a indica ca transferul e complet.

**5. (de prin cursul ILP si Superscalar)**

* **Se da urmatoarea secventa de cod: (28.01.2014)**

**add.d f0, f1, f8**

**add.d f2, f3, f8**

**add.d f4, f5, f8**

**add.d f6, f7, f8**

**Care din urmatoarele tehnici va aduce o imbunatatire pentru codul de mai sus:**

**-branch prediction**

**-out of order execution cu register renaming**

**-superscalar** ---> ce este asta? am gasit pe wikipedia ca e un CPU care implementeaza o forma de paralelism pe un singur procesor, executand mai multe instructiuni intr-un singur clock cycle pe diferite unitati de executie de pe CPU-ul respectiv.

Cred ca raspunsul este superscalar, pentru ca:

* branch prediction nu are cum sa fie, ca nu avem branchuri
* out of order execution cu register renaming nu ajuta la nimic, ca nu avem dependente de date si rearanjarea nu ar imbunatati nimic

Deci a ramas superscalar..si ar merge sa fie executate instructiunile in paralel din moment ce nu avem dependente de date.

* **Care dintre cele 3 abordari furnizeaza cea mai mare imbunatatire a performantei la rularea urmatorului cod: (08.09.2014) 03.02.2014)**

**LD R1 0(R2) #cache miss**

**ADD R2 R1 R1**

**LD R1 0(R3) #cache hit**

**LD R3 0(R4) #cache hit**

**ADD R3 R1 R3**

**ADD R1 R2 R3**

**Variante:**

* **out-of-order execution cu Register Renaming**
* **branch prediction**
* **superscalar**

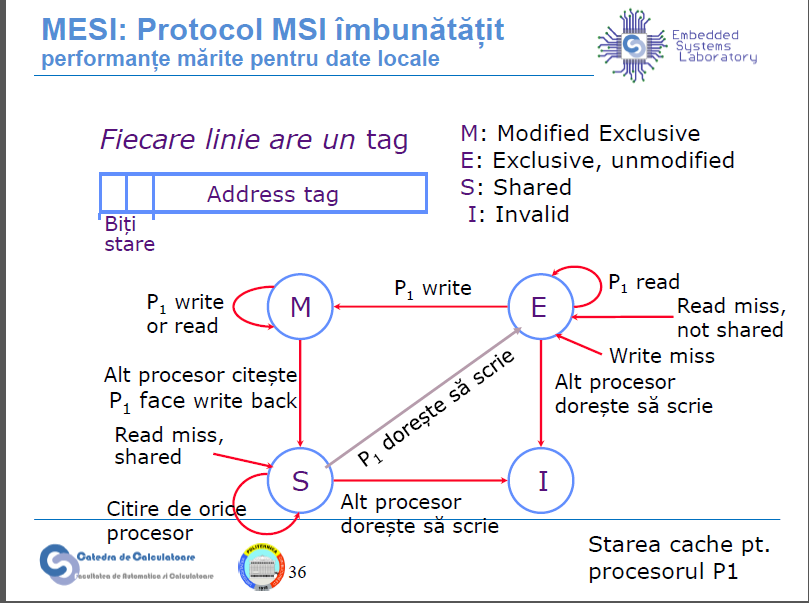
Banuiesc ca out-of-order execution, avand in vedere ca sunt dependente de date pe acolo. da, mai exact pentru ca avem WAW unde chiar e folosit register renaming.

**6. (de la multiprocesor)**

* **Definiti consistenta secventiala pentru un sistem multi-procesor. (08.09.2014) (28.01.2014)**

Un sistem este consistent secvential daca rezultatul oricarei executii e acelasi ca si cand operatiile tuturor procesoarelor ar fi executate secvential si operatiile fiecarui procesor individual apar in ordinea specificata in program. (Leslie Lamport)

* **Descrieti protocolul MESI. (03.02.2014) - e cam messy stiu cred ca Danut trebuie sa aiba o zi proasta ca sa o dea pe asta**



**In** [**computing**](https://en.wikipedia.org/wiki/Computing)**, the MSI protocol - a basic** [**cache-coherence protocol**](https://en.wikipedia.org/wiki/Cache_coherence) **- operates in multiprocessor systems. As with other cache coherency protocols, the letters of the protocol name identify the possible states in which a cache line can be. So, for MSI, each block contained inside a cache can have one of three possible states:**

* **Modified: The block has been modified in the** [**cache**](https://en.wikipedia.org/wiki/CPU_cache)**. The data in the cache is then inconsistent with the backing store (e.g.** [**memory**](https://en.wikipedia.org/wiki/Main_memory)**). A cache with a block in the "M" state has the responsibility to write the block to the backing store when it is evicted.**
* **Shared: This block is unmodified and exists a read-only state in at least one cache. The cache can evict the data without writing it to the backing store.**
* **Invalid: This block is not present in the current cache, and must be fetched from memory or another** [**cache**](https://en.wikipedia.org/wiki/CPU_cache) **if the block is to be stored in this cache.**

**When a read request arrives at a cache for a block in the "M" or "S" states, the cache supplies the data. If the block is not in the cache (in the "I" state), it must verify that the line is not in the "M" state in any other cache. Different caching architectures handle this differently. For example, bus architectures often perform** [**snooping**](https://en.wikipedia.org/wiki/Bus_snooping)**, where the read request is broadcast to all of the caches. Other architectures include** [**cache directories**](https://en.wikipedia.org/wiki/Directory-based_coherence_protocols) **which have agents (directories) that know which caches last had copies of a particular cache block. If another cache has the block in the "M" state, it must write back the data to the backing store and go to the "S" or "I" states. Once any "M" line is written back, the cache obtains the block from either the backing store, or another cache with the data in the "S" state. The cache can then supply the data to the requestor. After supplying the data, the cache block is in the "S" state.**

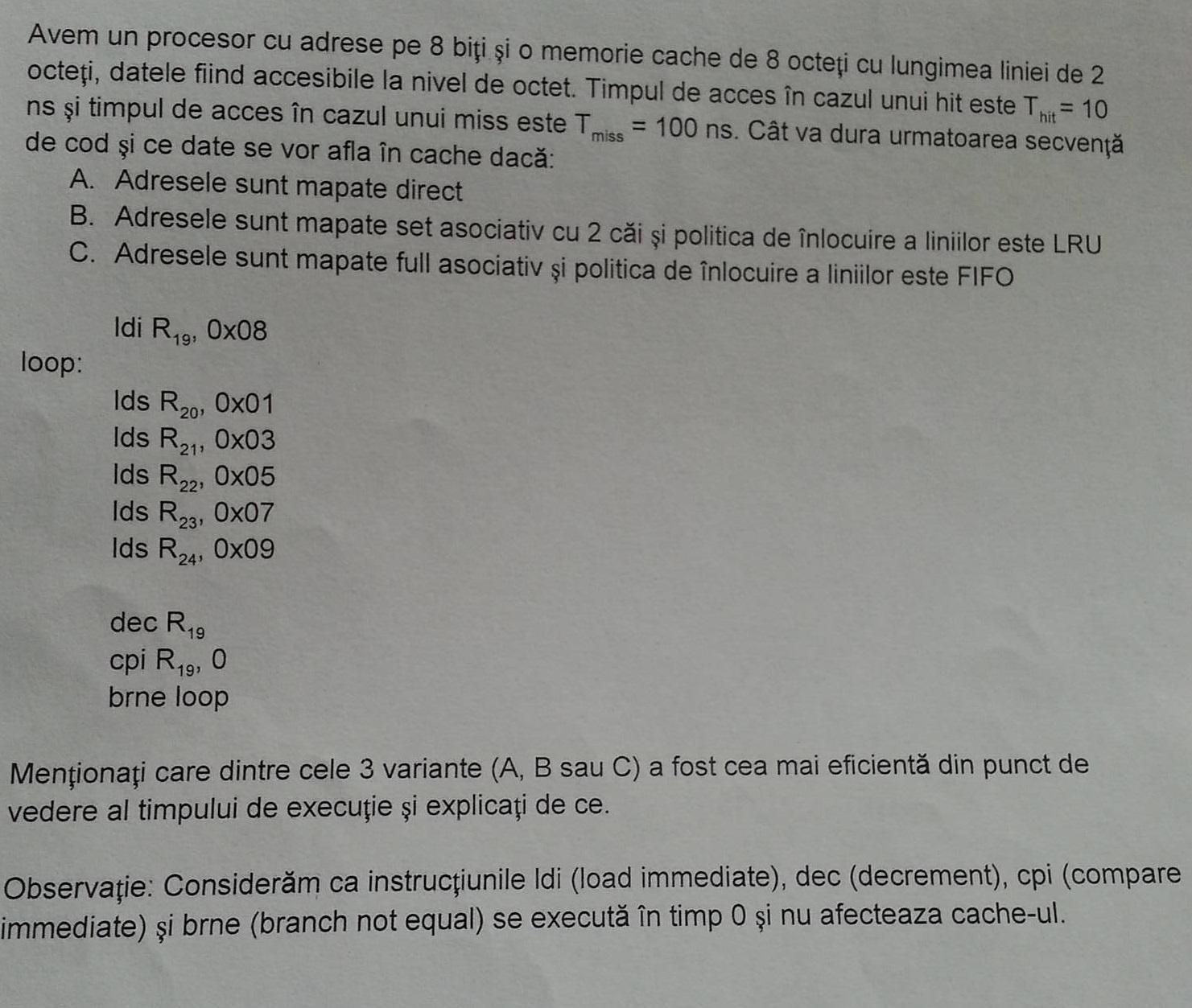
**When a write request arrives at a cache for a block in the "M" state, the cache modifies the data locally. If the block is in the "S" state, the cache must notify any other caches that might contain the block in the "S" state that they must evict the block. This notification may be via bus snooping or a directory, as described above. Then the data may be locally modified. If the block is in the "I" state, the cache must notify any other caches that might contain the block in the "S" or "M" states that they must evict the block. If the block is in another cache in the "M" state, that cache must either write the data to the backing store or supply it to the requesting cache. If at this point the cache does not yet have the block locally, the block is read from the backing store before being modified in the cache. After the data is modified, the cache block is in the "M" state.**

**7. (de la Dan)**

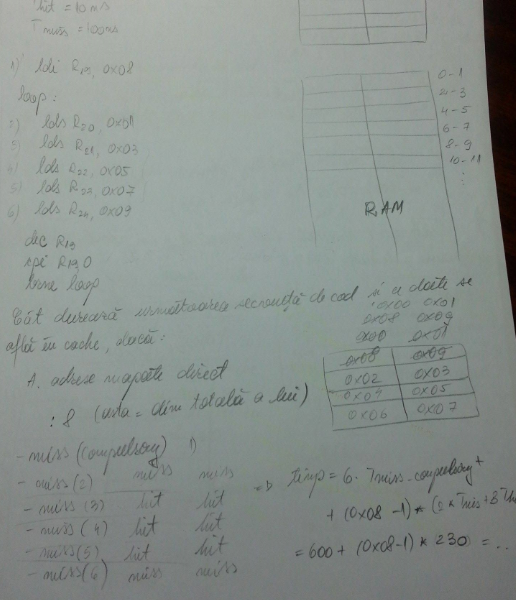
* **Care sunt principalele caracteristici ale procesorului pe care l-ati folost la lab? (10.02.2015)**
* arhitectura RISC
* banda de asamblare in 5 etape
* registri pe 8 biti
* 32 de registri generali ( dar se foloseau doar 16, nu?) - noi am folosit doar 16 pentru ca multe instructiuni folosesc doar 16, exista instructiuni care pot sa ii foloseasca si pe ceilalti 16
* adresare pe 8 biti

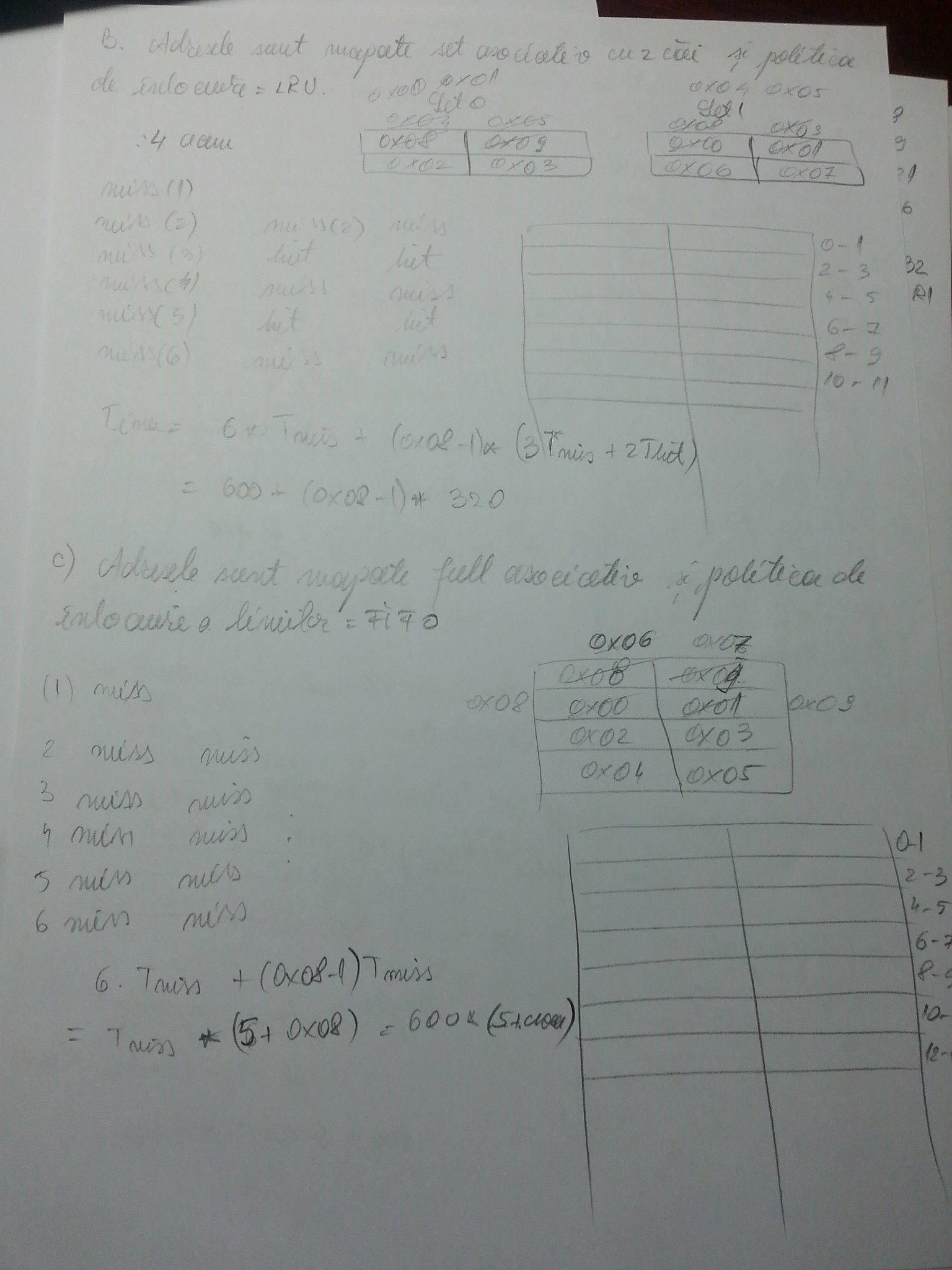
**Partea 2 (1p --- cica, nu stiu daca este asa cu adevarat)**

(10.02.2015)

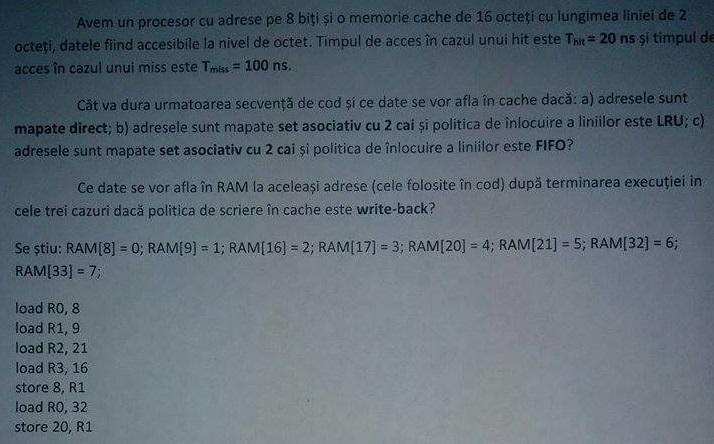


la primul ldi nu este miss, greseala mea :(

-



(08.09.2014)

****

(03.02.2014)

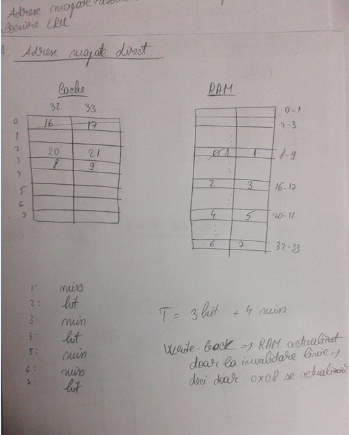
ATENTIE!!!

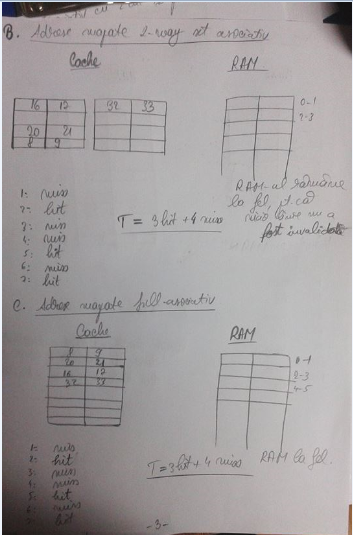
**8-9 se mapeaza la 4**

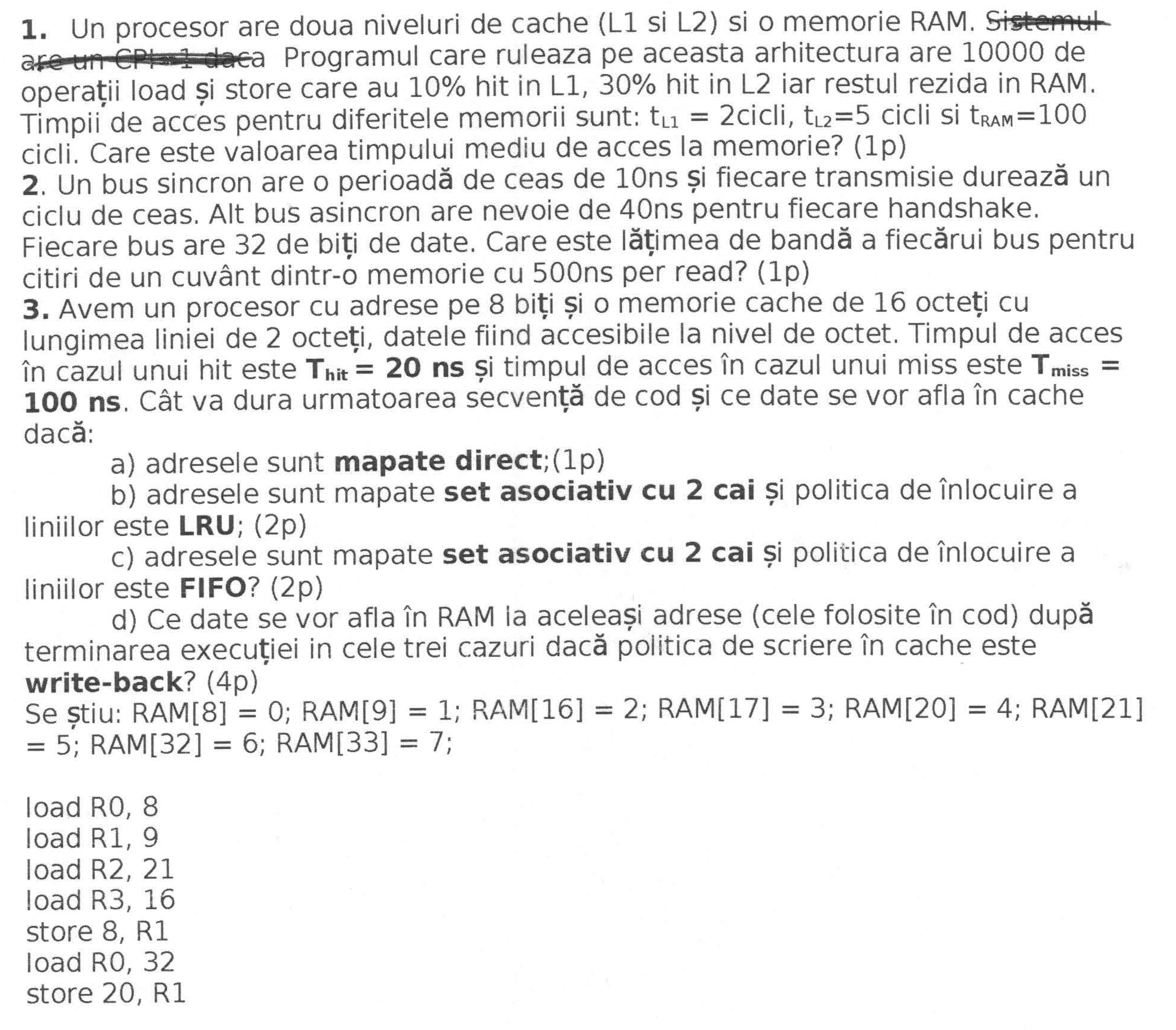
|  |  |  |
| --- | --- | --- |
|  |  |  |

**4 e tot miss, la 5 ar fi trebuit sa fie hit**

**RAM-ul nu cred ca e bine actualizat, pentru ca la write-back se actualizeaza doar linia invalidata, nu tot ce are flag “dirty” pana atunci.**







1. E o combinatie intre AMAT si niveluri multiple de cache, dar nu stiu sigur cum se face

(am urmat modelul din curs)

2. Pentru bus sincron:

* trimite adresa la memorie: 10ns
* citeste memoria: 500ns
* trimite date la dispozitiv: 10ns

=> in total = 500 + 10 + 10 = 520ns

=> BW = 4 octeti la 520ns => 4 / 520 = 7.69 MB/s

Pentru bus asincron:

* memory read: 40ns
* data ready & hadshake: 3 \* 40 + 500 = 620 ns
* read & ack: 3 \* 40 = 120 ns

=> in total = 780 ns

=> BW = 4 octeti la 780ns => 4 / 780 = 5.12 MB/s

3. e acelasi cu cel de mai sus

(28.01.2014)

Este partea de problema care consta din 3 exercitii revin cu ele cand fac rost

In principiu erau probleme cu speedup, performanta si numarat miss-uri intr-un cache.

**Rezolvare of Problemă examen CN2 09.02.2015**

Avem un procesor cu adrese pe 8 biți și o memorie cache de 16 octeți cu lungimea liniei de 2 octeți, datele fiind accesibile la nivel de octet. Timpul de acces în cazul unui hit este Thit = 50 ns și timpul de acces în cazul unui miss este Tmiss = 250 ns. Cât va dura urmatoarea secvență de cod și ce date se vor afla în cache dacă:

1. Adresele sunt mapate direct
2. Adresele sunt mapate set asociativ cu 2 căi și politica de înlocuire a liniilor este LRU
3. Adresele sunt mapate full asociativ și politica de înlocuire a liniilor este FIFO

Ce date se vor afla în RAM la aceleași adrese (cele folosite în cod) după terminarea execuției in cele trei cazuri dacă politica de scriere în cache este write-through?

Se știu:

RAM[0x04] = 0;

RAM[0x05] = 1;

RAM[0x0A] = 2;

RAM[0x0B] = 3;

RAM[0x1A] = 4;

RAM[0x1B] = 5;

RAM[0x40] = 6;

RAM[0x41] = 7;

1. lds R19, 0x0A
2. lds R20, 0x0B
3. sts 0x40, R20
4. lds R20, 0x1A
5. lds R21, 0x05
6. sts 0x05, R19
7. lds R19, 0x0A

Observație: Pentru fiecare subpunct scrieți ce conține memoria RAM la adresele folosite în cod, ce se află în fiecare linie de cache la finalul execuției codului și timpul total de execuție. Puteți scrie timpul necesar fiecărei instrucțiuni (se punctează parțial).

lds R19, 0x0A ; miss

lds R20, 0x0B ; hit

sts 0x40, R20 ; miss

lds R20, 0x1A ; miss

lds R21, 0x05 ; miss

sts 0x05, R19 ; hit

lds R19, 0x0A ; miss

Cache:

0: 0x40, 0x41

1: x, x

2: 0x04, 0x05

3: x, x

4: x, x

5: ~~0x0A~~ ~~0x1A~~ 0x0A, ~~0x0B~~ ~~0x1B~~ 0x0B

6: x, x

7: x, x

RAM:

RAM[0x40] = 3;

RAM[0x05] = 2;

T = 5\*Tmiss + 2\*Thit = 1350

1. b

lds R19, 0x0A ; miss

lds R20, 0x0B ; hit

sts 0x40, R20 ; miss

lds R20, 0x1A ; miss

lds R21, 0x05 ; miss

sts 0x05, R19 ; hit

lds R19, 0x0A ; hit

Cache:

0.0: 0x40, 0x41

0.1: x, x

1.0: 0x0A, 0x0B

1.1: 0x1A, 0x1B

2.0: 0x04, 0x05

2.1: x, x

3.0: x, x

3.1: x, x

RAM:

RAM[0x40] = 3;

RAM[0x05] = 2;

T = 4\*Tmiss + 3\*Thit = 1150

1. c

lds R19, 0x0A ; miss

lds R20, 0x0B ; hit

sts 0x40, R20 ; miss

lds R20, 0x1A ; miss

lds R21, 0x05 ; miss

sts 0x05, R19 ; hit

lds R19, 0x0A ; hit

Cache:

0: 0x0A, 0x0B

0: 0x40, 0x41

0: 0x1A, 0x1B

0: 0x04, 0x05

0: x, x

0: x, x

0: x, x

0: x, x

RAM:

RAM[0x40] = 3;

RAM[0x05] = 2;

T = 4\*Tmiss + 3\*Thit = 1150

:p